Integration of a Vehicle Operating Mode Management into UNICAR*agil*'s Automotive Service-oriented Software Architecture

Inga **Jatzkowski**; Torben **Stolte**; Robert **Graubohm**; Prof. Dr.-Ing. Markus **Maurer**
Institute of Control Engineering, Technische Universität Braunschweig, Braunschweig, Germany

Alexandru **Kampmann**; Dr.-Ing. Bassam **Alrifaee**; Prof. Dr.-Ing. Stefan **Kowalewski**
RWTH Aachen University, Aachen, Germany

Dr.-Ing. Michael **Buchholz**; Prof. Dr.-Ing. Klaus **Dietmayer**
Institute of Measurement, Control and Microtechnology, Ulm University, Ulm, Germany

Contact: jatzkowski@ifr.ing.tu-bs.de

Content

Summary

Automated vehicles require a central decision unit in order to coordinate the responsibility for the driving task between multiple operating modes. Additionally, other non-driving related tasks such as operation of an automatic door system must be coordinated as well. In this paper, we will motivate the usefulness of such a central decision unit at the example of the operating mode management of the UNICAR*agil* project. We will describe its integration with UNICAR*agil*'s Automotive Service-oriented Software

https://doi.org/10.24355/dbbs.084-202110271613-0

Architecture and how modularity of this service-oriented software architecture is ensured. An example from the project's context will further illustrate the functioning principle of the operating mode management in combination with the service orchestration of the Automotive Service-oriented Software Architecture.

1 Introduction

The automated vehicles developed in the project UNICAR*agil* can be operated in four different operating modes: *automated operation*, *control center operation*, *safe halt*, and *manual operation* [1]. In automated operation mode, which is the default mode of the vehicles, the vehicles are supposed to handle all traffic situations and challenges that occur in operation. Control center operation is a fallback solution in the event that the automated vehicle guidance system is unable to resolve a situation on its own that has occurred in the open world. Safe halt is a fallback solution to cope with severe degradation and initiate a transition into a minimal risk condition [2]. Manual operation is only enabled for maneuvering the vehicle at very low speeds, e.g., in a vehicle repair shop.

An automated vehicle with several different operating modes requires a central decision unit to coordinate mode changes. Any mode transition requires a set of conditions to be fulfilled. For example, the mode automated operation must not be activated when necessary components are not available. Software components in the UNICAR*agil* vehicles are realized as services as the UNICAR*agil* vehicles feature the Automotive Service-oriented Software Architecture (ASOA) [3], including a service orchestration. This service orchestration has the ability to start and stop services and to connect service interfaces. This approach allows for a reconfiguration of the set of active services without losing the benefit of modularity of a service-based software architecture. Each operating mode requires a specific configuration of services. The operating mode management interacts with the ASOA's service orchestration. Services are kept unaware of the vehicle's operating mode and are activated or deactivated by the ASOA's Orchestrator depending on the current operating mode.

As stated above, one main task of such an operating mode management is to coordinate the responsibility regarding the interfacing with actuation systems performing the driving task. Additionally, tasks unrelated to the driving task itself must also be coordinated in a fully automated vehicle. Examples for non-driving related tasks are opening the vehicle's automatic doors, calling the control center, or activating and deactivating the lights. In a manually operated car, the driver is responsible that the vehicle's doors are not opened while driving or when only stopping temporarily (e.g., at a red traffic light). In an automated vehicle, on the other hand, it is the vehicle's responsibility that the doors are only opened at appropriate stops.

Thus, a central decision unit like an operating mode management in an automated vehicle has two main tasks: to coordinate the responsibility for the driving task using operating mode changes and to coordinate non-driving related tasks.

https://doi.org/10.24355/dbbs.084-202110271613-0

In this contribution, after a brief overview of related work regarding operating mode management systems, we present the role of the different operating modes in UNICAR*agil*'s ASOA and the project's automation concepts. Following, we will present in detail the operating modes and the interaction with the ASOA's service Orchestrator and illustrate their interaction at an example. This paper serves as an initial overview of the approach the authors currently develop in the project. It depicts an early stage of development and shall be built upon in future publications.

2 Related Work

As soon as a mobile agent offers diverse control options, a concept for distinguishing and managing operating modes becomes necessary. A rigorous analysis of the state of the art has not yet accomplished. At the moment, we only refer to selected previous work from different fields of application which describe mode transitions between automated operation and teleoperation of the agent. For example, the fundamental need for a control mode manager to coordinate between remote and automatic control of robots, as well as an illustration of architectural consequences, can already be found in [4], where the concept of remote robot teleoperation is discussed.

Bodell and Gulliksson [5] describe the design and simulation of a system for teleoperation of an autonomous heavy duty truck. The remote control of the vehicle includes following pre-planned paths autonomously in addition to being controlled by a human. The authors design their system to allow for spontaneous human operator takeover at any time by performing Autonomous Synchronization with a System Coordinator node that is part of the control center's system architecture. In Fig. 1, Bodell and Gulliksson illustrate the modes of operation for the system under development. Autonomous Safe (AS) is the automated navigation, Manual Safe (MS) is the operating mode of teleoperation with an automatic emergency braking system being active, and Manual Unsafe (MU) is the operating mode of teleoperation with overridden emergency braking system (e.g., to cope with sensor malfunction).

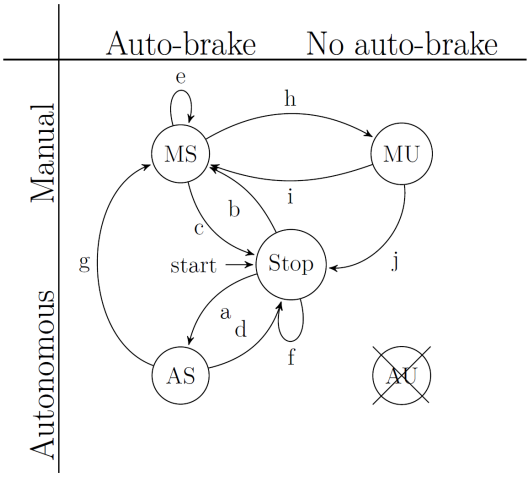

Fig. 1 Control modes and autonomous synchronization by Bodell and Gulliksson with modes: Autonomous Safe (AS), Manual Safe (MS), Manual Unsafe (MU), Autonomous Unsafe (AU). ([5], p.40).

https://doi.org/10.24355/dbbs.084-202110271613-0

In the context of urban search and rescue robots that allow for both autonomous and teleoperated navigation, Bruemmer et al. [6] introduce four modes of operation that allow their robot to adjust its level of autonomy on the fly: Teleoperation, Safe Mode, Shared Control, and Full Autonomy. A more detailed system description, which also addresses ergonomic aspects, can be found in [7]. However, the authors do not share details on their system's mechanisms for operating mode management.

Further related work on the subject of operating mode management describes systems based on conventional, manually controlled land vehicles. One example is the work of Appelqvist et al. [8] in which a common all-terrain vehicle is the basis of a demonstrator for military unmanned ground vehicle applications. The state-machine presented by the authors, which we show in Fig. 2, accordingly assumes a basic state of manual vehicle guidance and human supervision, which significantly limits its applicability to the UNICAR*agil* research project.

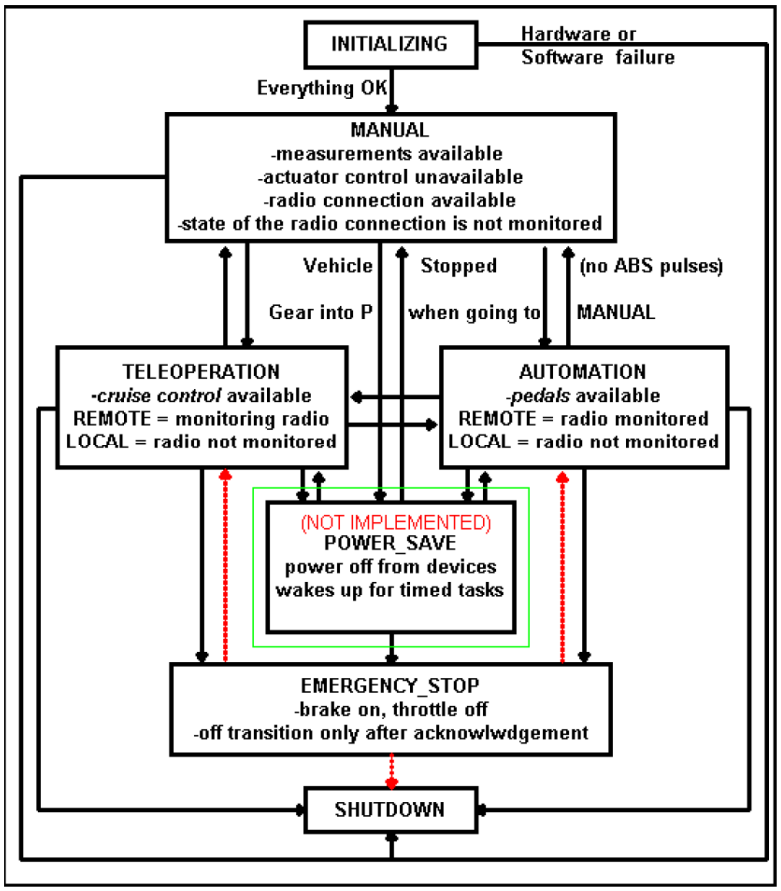

Fig. 2 Driving computer state machine ([8], p. 250).

3 Role of operating modes in ASOA

The software running in the UNICAR*agil* vehicles uses the Automotive Service-oriented Software Architecture (ASOA) [3]. We will now give a brief overview of ASOA and explain the role of the operating mode management in the context of ASOA.

https://doi.org/10.24355/dbbs.084-202110271613-0

To understand the relevance of various concepts in ASOA, we will first describe prevalent architectures in today's production vehicles. During the vehicle engineering process, dozens of subsystems – often provided by numerous suppliers – are integrated to constitute the final vehicle. The integration often occurs at an electronic control unit (ECU) level and many subsystems often come with their own custom-tailored ECU. During the design time, the resulting architectures are rigidly integrated and are not expected to change significantly after the design process. More specifically, the interplay between different ECUs and their software components is fixed and typically does not change for the lifetime of the vehicle model [9]. Individual components have information about the overall system integration embedded into them. This may include embedding technical information about the sources of specific information (e.g., Control Area Network (CAN) IDs) or the reliance of components on information about internal states of other components. Consequentially, the overall system architecture becomes difficult to update and to reconfigure after completion of the engineering process. As system integration information is leaked into individual components, updating, replacing, or reconfiguring specific parts of the system quickly requires updating many components. Furthermore, due to the often unclear functional interdependence between components – residing in the heads of engineers and in design documents rather than the software itself – foreseeing all consequences of updates and system reconfigurations becomes even harder [10].

These unforeseeable consequences do not necessarily pose a problem for current road vehicles, as updates rarely add new functions, with Tesla being a notable exception. We argue that the ability to update the software architecture becomes critical for the implementation of future automated and connected vehicles. For automated vehicles, updatability allows vehicles that have been produced and shipped to customers to always operate using the state-of-the art techniques in perception and control. This is especially important from a safety perspective, as passenger's safety in automated vehicles is even more in the hands of the software. The aspect of safe updates of vehicle functions after vehicle deployment was a central research focus of the project Controlling Concurrent Change [11] and will only gain relevance in the future.

The ASOA has been devised around several core objectives that allow for updatability of the software architecture. Following a service-oriented approach (SOA), the fundamental building blocks are modular services that are dynamically integrated at runtime. Dynamic integration means that the interplay between individual services is decided at runtime. For example, as the current vehicle speed may be required by a service, the specific service providing the vehicle speed signal is not hard-coded but instead selected at runtime. For this purpose, ASOA services follow a unified specification approach for requested and offered interfaces. This information constitutes the digital datasheet of a service. As service interfaces are explicitly announced and discovered at runtime, clear insight into functional interdependence is achieved. Due to the runtime integration and the interface concept, services themselves and, thus, the overall software architecture become modular.

The Orchestrator is a designated architecture component in ASOA and is the only element that can control the integration between individual services. Among others, the

https://doi.org/10.24355/dbbs.084-202110271613-0

Orchestrator has mechanisms to change integration between services by linking interfaces and has the ability to activate and deactivate services. The Orchestrator itself tightly integrates with the operating mode management. While services themselves are kept modular, the operating mode management is the central entity that controls the current mode of operation. This entity bundles global knowledge about the different operating modes, the intended system integration in each operating mode as well as the transition between modes of operation. The operating mode management uses the mechanisms offered by the Orchestrator to establish the intended mode of operation. More specifically, the Orchestrator offers an Application Programming Interface (API) that allows to link service interfaces, to control the lifecycle of individual services, and to determine service liveliness.

In the design process of the services in UNICAR*agil*, specific attention was paid to avoid exposing global state information to individual services. A simple example for this is the door management service. For the door management to determine whether it is safe to open the door, it is insufficient to decide simply based on the vehicle's motion. The vehicle may be stopped because it is waiting at a red light or because the vehicle reached its destination. In a naive approach, the door management service would have to explicitly know about internal states of services that may only be available in specific vehicle variants. This in turn makes the door management service hard to use in other contexts. In our approach, the decision whether it is safe to open the door or not is not made in the door management service, but on the level of the operating mode management. Here, all necessary information is available without having to pollute the door management service with global state information. Thus, the door management becomes a modular service that offers a simple interfaces to control the vehicle door and is seamlessly reusable across different vehicle variants.

Our approach contrasts with prevalent architectures, where individual components require knowledge about the overall system state. Knowledge about the system's blueprint and the individual building blocks constituting the system is clearly separated. We argue that this approach has various advantages. First, individual services can be reused in different vehicle variants and contexts that have different operating modes. Second, changing the system's blueprint can be done in a more agile manner, as individual services do not have to be modified to be used in different operating modes. This is especially important for keeping up with short technology lifecycles. If the mode of operation was embedded in each individual service, adding new modes of operation would require updating all services that rely on knowledge about the global system state. Our approach contributes to the overall updatability of the software architecture and allows us to handle agility in the UNICAR*agil* project. Finally, through the strict separation of concerns, overall system modularity is facilitated.

https://doi.org/10.24355/dbbs.084-202110271613-0

4 Operating mode management

To illustrate the operating mode management, we will first give an overview of the different operating modes of the UNICAR*agil* vehicles and explain their use. Afterwards, we will elaborate how the operating mode management functions and illustrate this at an example.

4.1 The operating modes of the UNICAR*agil* vehicles

As stated above, the UNICAR*agil* vehicles can be operated in four different operating modes regarding the driving task: *automated operation*, *control center operation*, *safe halt*, and *manual operation*. *Automated operation* is the default operating mode. In this mode, the dynamic driving task is performed fully by the vehicle's automated driving system. All monitoring tasks are performed by the vehicle as well as non-driving related tasks.

In order to facilitate automated operation, all services required for automated guidance must be active and all necessary interfaces must be connected properly. All systems must be operational and calibrations must be complete.

Control center operation [12] is the second operating mode of the UNICAR*agil* vehicles. It allows a vehicle to be controlled via teleoperation. Teleoperation is realized in different forms. A trajectory generated by the control room can be transmitted to the vehicle and realized by the vehicle's trajectory controller. Alternatively, low-level driving commands can be submitted to the vehicle and realized directly by the vehicle's dynamic modules (a combination of wheel hub motor, steering system, and brake system, c.f. [12]). Manipulation of environment perception data was also considered but not realized within the project UNICAR*agil*, e.g., to reclassify an object in the vehicle's path if the object was misclassified by the vehicle's perception system. The addition of a control center can enhance availability of an automated vehicle by increasing the number of situations an automated vehicle can handle as unforeseeable events will always happen in an open world. Germany's draft law "Act on Autonomous Driving" [13] designed to amend the Road Traffic Act also formulates the need for a control center for vehicles in which no human driver is present.

In order to operate a vehicle remotely from the control center, a connection between vehicle and control center must be established. The vehicle's perception data must be rerouted to the control center for the human operator in the control center to assess the current situation of and around the vehicle. Depending on the teleoperation mode, the vehicle must be reconfigured to accept control inputs from the control center.

In case the vehicle encounters major system degradation, *safe halt* mode is activated. In safe halt mode, the vehicle transitions to a minimal risk condition by performing a minimal risk maneuver [2], [14]. The detection of such a degradation is part of a self-perception system. Self-perception describes the counterpart to environment perception. While environment perception provides information on the environment external to the vehicle, self-perception provides information on the internal state of the vehicle

https://doi.org/10.24355/dbbs.084-202110271613-0

[15]. In case of the UNICAR*agil* vehicles, the minimal risk condition is basically a standstill at an appropriate location. The vehicles are designed for urban environment where stopping the vehicle at the side of the road outside of intersections or other high-traffic areas can be considered reasonably safe in most cases even when the passengers have to exit the vehicle. This may not be the case for automated vehicles operated on a highway where stopping on the shoulder of the road may be the minimal risk condition, but cannot generally be considered to maintain the safe state [2].

The needs of different groups of passengers of an automated vehicle and the resulting requirements of these passengers in case of an unplanned safe halt are discussed in [16].

Once safe halt mode is activated, the control center is alerted. When the minimal risk condition has been reached, the control center is required to take action. As safe halt mode is only activated when the regular automated driving function is unable to safely fulfill the driving task any longer, an assessment of the degradation by an external operator in a control center is indispensable. Depending on the system degradation of the vehicle, support provided by the control center can take different forms. If the system degradation was only temporary, acknowledging safe halt activation and triggering a transition back into automated operation can be sufficient. If the present degradation makes it impossible to proceed the trip, but slow teleoperation is still possible, the control center operator can also request control of the vehicle in order to move the vehicle to a more appropriate location. Finally, the remote operator can send assistance to the vehicle if control via teleoperation is not possible.

Safe halt mode and its accordingly activated safe halt function are safety fallbacks. Thus, transition into safe halt mode does not depend on the fulfillment of any conditions and should be available permanently during operation time. In the UNICAR*agil* vehicles, the safe halt function uses a sensor setup redundant to and independent from the sensor modules at the four corners of the vehicles [1], [12]. This redundant sensor setup is used to check for objects in the immediate path of the vehicle. Its function is intentionally kept simple to minimize complexity in the safety fallback. Safe halt should be used sparingly and only in emergencies due to this minimal sensor setup and reduced functionality of the safe halt function compared to the regular automated driving function [14].

The last operating mode is *manual operation*. Automated vehicles also infrequently need to be moved in distinct environments outside of the road network and common parking grounds. One example is to undergo maintenance as any other vehicle. Here, it may become necessary to manually move the vehicle in a workshop. It is not practical to include this use-case in the automated driving system as it is rare and would have to cover a large variety of situations. Therefore, manual operation was included as an operating mode for the UNICAR*agil* vehicles in order to move a vehicle at very low speed for maintenance purposes in a workshop.

The UNICAR*agil* vehicles are designed without any manual control inputs for a driver. Thus, in order to activate manual operation, manual control elements must be connected to input sockets provided for this purpose. While these control elements are

https://doi.org/10.24355/dbbs.084-202110271613-0

connected, automated operation is not possible. In addition to the manual control elements, services that translate manual control inputs into actuator commands must be activated.

4.2 Switching between Operating Modes

Connections and transitions between the operating modes described in the previous section are illustrated in Fig. 3. Transitions between modes always take place in standstill for safety reasons. The only exception to this are transitions into safe halt mode, as this mode encapsulates the safety fallback of the automated driving system.

After startup, the vehicle enters a boot-up state ①, Boot-Up is one of several so called transition states in order to connect the operating mode management with UNICAR*agil*'s Automotive Service-oriented Software Architecture (ASOA) [3]. Within these transition states, the service orchestration takes place. Thus, necessary services are activated, service interfaces are linked, and conditions for transitioning into one of the operating modes are evaluated. In case service orchestration is unsuccessful, the operating mode management transitions back into the previous operating mode and another mode change can be initiated.

In the boot-up state, all necessary services for automated operation are started, inputs and outputs are connected, and system calibrations are carried out. If no manual control elements are connected to the vehicle and all conditions for automated operation are fulfilled, the operating mode management transitions (a) into automated operating mode ⑤. If any manual control elements are connected to the vehicle, the operating mode management initiates the transition (b) into manual operating mode ③. Another transition state ② may be necessary as all services are configured for automated operation during boot-up and may have to be reconfigured to accept inputs from the manual control elements.

The removal of all manual control elements triggers the transition (d) into automated operation ⑤. Again, a transition state ④ is necessary similar to the boot-up state to start all required services, connect services with each other, and check whether all conditions for a transition (e) into automated operation are fulfilled.

From automated operation ⑤, a transition (g) into control center operation ⑦ can be triggered, e.g., when the vehicle cannot resolve a situation on its own. As the control center requires access to the vehicle's perception data, a reconfiguration of the perception system may be necessary. Additional services are started in a transition state ⑥ for connecting the vehicle directly to the control center, for sending data to the control center, and for receiving data from the control center. Depending on the method of teleoperation, services for trajectory control or actuator control are reconfigured to receive input from the control center.

Transitioning (i) back to automated operation ④ from control center operation requires all the aforementioned reconfigurations to be reversed. The perception system must be reconfigured for automated operation, all control inputs must be reconnected to the vehicle's internal trajectory planner or trajectory controller. All services for connecting

https://doi.org/10.24355/dbbs.084-202110271613-0

with the control center must be stopped and the connection to the control center must be terminated.

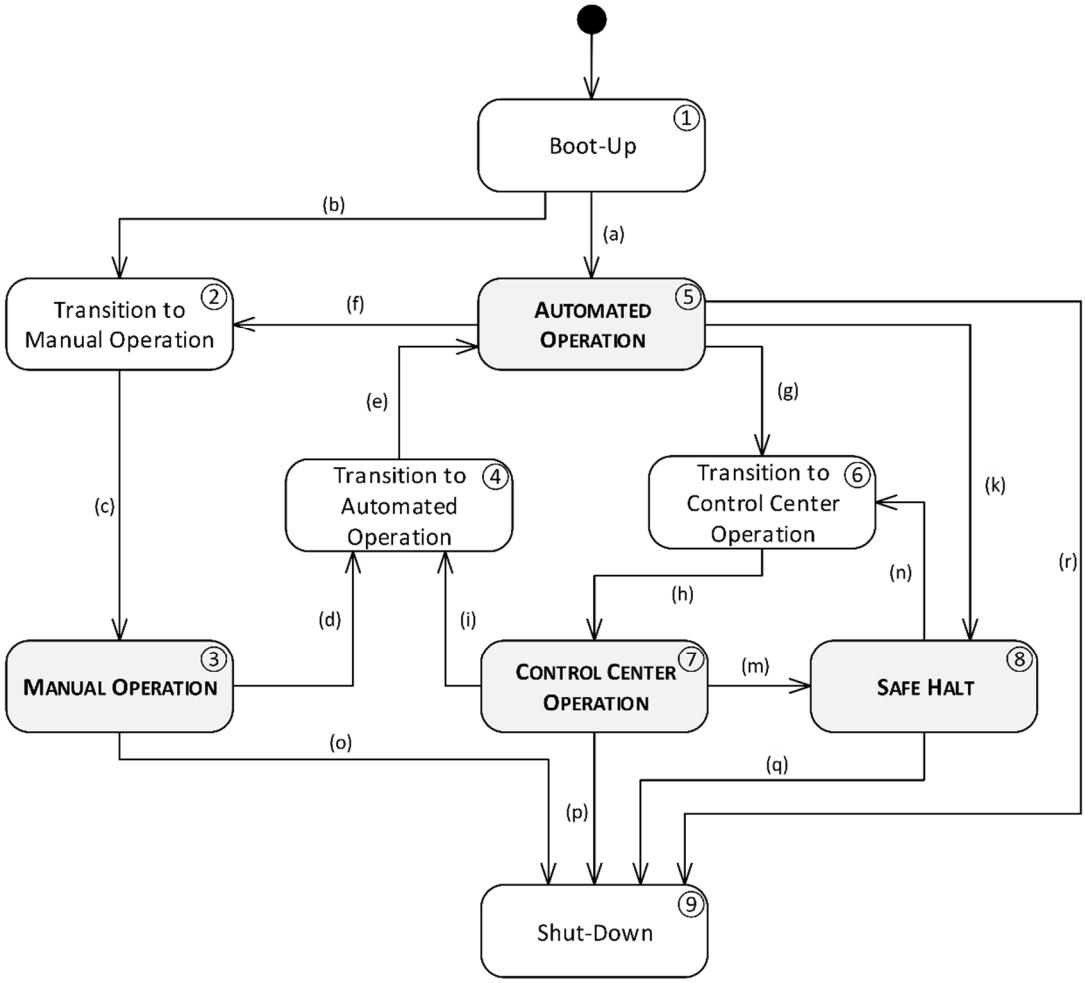

Fig. 3 Overview of operating modes and mode transitions.

From automated operation as well as from control center operation, transitioning via (k) and (m), to safe halt mode ⑧ is possible without any transition state. As stated above, in safe halt mode, the safe halt function is activated which acts as the fallback for the automated driving system. This transition is only activated when major system degradation occurs. Thus, it is essential that a transition into safe halt mode is fast and not delayed by any transition state or conditions. Transitioning back to automated operation from safe halt mode is only possible via control center operation. As a major system degradation occurred it is necessary that the reason for this degradation is assessed before automated operation is continued.

When the vehicle shall be turned off, the transition state Shut-Down ⑨ is entered (o), (p), (q), (r). This transition state allows for all services to be stopped and all systems to be shut-down orderly before power to the systems is disconnected.

https://doi.org/10.24355/dbbs.084-202110271613-0

5 Implementation

This section provides details on the implementation of the operating mode management and its integration with ASOA and the Orchestrator.

As describe before, the operating mode management is an automaton consisting of states that are linked with transitions. Transitions are executed if Boolean-value guard conditions are true. When executing a transition or when entering a state, the automaton may invoke specific functions offered by the Orchestrator. For UNICAR*agil*, we use MATLAB Stateflow for implementation of the operating mode management, as it has all features required for implementing automata as described above. Guards and actions that interact with the Orchestrator and ASOA are functions implemented using C language, which can be directly invoked from Stateflow. The complete operation mode management for all four vehicle variants resides in a single MATLAB Stateflow chart. We use the MATLAB code generator to create a binary that can be directly deployed to the UNICAR*agil* vehicles and git for version control. The deployment and code generation process is automated using continuous integration (CI). Upon changing the operation mode management, the CI pipeline automatically generates a binary that is ready to run on the target platform. For debugging and development purposes, the model can also be executed in the MATLAB environment.

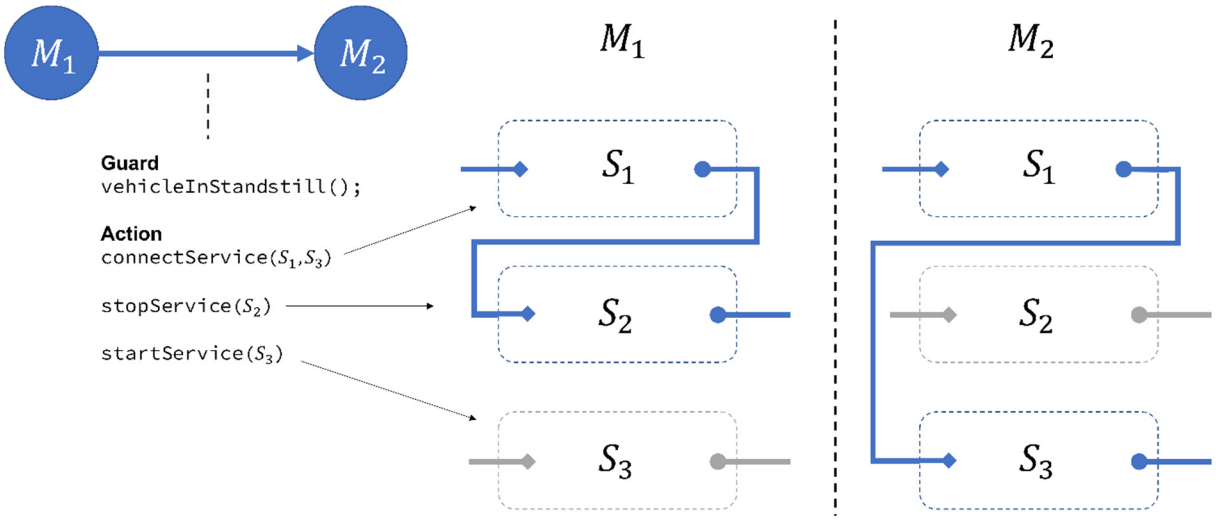

Fig. 4 ASAO implementation example for a simplified mode change.

A simplified example for such a mode change is depicted in Fig. 4. Here, the operating mode management is currently in mode M_1. The only outbound transition from M_1 leads to M_2, but is only executed if the guard condition becomes true. The guard condition here is implemented as the C function `vehicleInStandstill()`, which evaluates true if the vehicle is not in motion. Once this condition becomes true, a sequence of commands (`connectService`, `stopService`, `startService`) is issued using the Orchestrator. These actions change the set of active services and the integration among them. The orchestrator commands are received and handled internally by the services and are invisible to the developer of the service. First, the output of service S_1 is linked to the input of service S_3. Then, service S_2 is stopped and service S_3 is started. The operating mode management has now transitioned to mode M_2. Note that through

https://doi.org/10.24355/dbbs.084-202110271613-0

the decoupling of individual services from knowledge about the overall system integration, services do not have to be aware of the higher-level operating mode management. Thus, services become reusable in other contexts and adding or changing the system integration behavior does necessarily require updates of individual services.

6 Example

In the following, we will illustrate the mechanisms described above at an example. The example is a fallback to safe halt from automated operation, resolved by the control center with subsequent continuation of the trip.

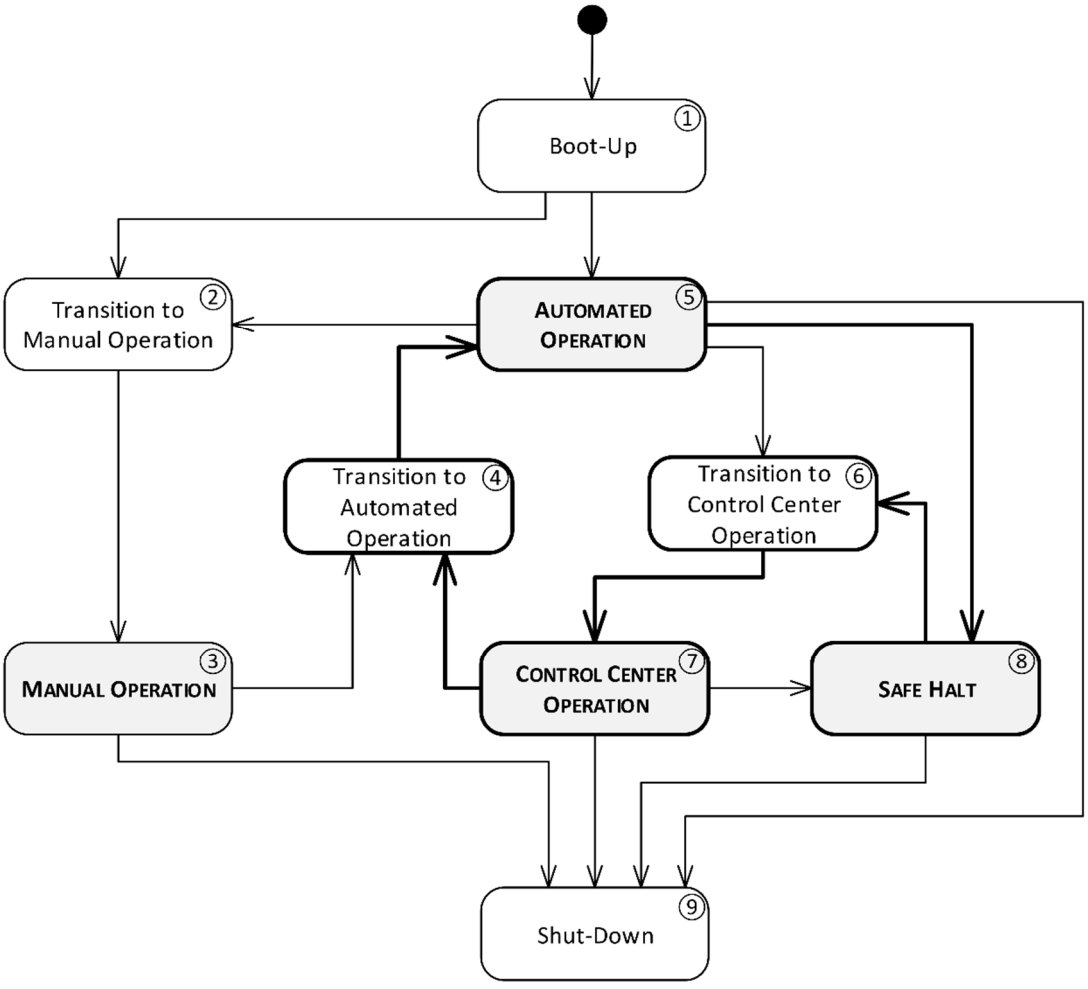

Fig. 5 Overview of operating modes and transitions with example bolded.

6.1 Transition from automated operation into safe halt mode

We will begin the example with the system in the operating mode automated operation, as it is the default operating mode the vehicle will reside in the majority of the time. From automated operation, a transition into safe halt mode is only triggered as a last resort. As stated in Section 4, a transition into safe halt mode and the connected safe halt functionality is only triggered when a severe degradation occurs and the automated driving function active in automated operating mode is not able to safely fulfill

https://doi.org/10.24355/dbbs.084-202110271613-0

the driving task any longer. Such a degradation can be caused by, e.g. a severe sensor degradation, loss of essential ECUs, or degradation or loss of other essential services or components.

Information provided by the vehicle's self-perception is used in the operating mode management in the transitional condition to trigger a transition into safe halt mode and activate the safe halt function, which performs a minimal risk maneuver in order to reach a minimal risk condition [2], [14]. The safe halt function is realized as a service in hot standby that is constantly provided an emergency path by the trajectory planner of the main automated driving function. This path is constantly transformed into a trajectory by the safe halt function and sent to the trajectory controller.

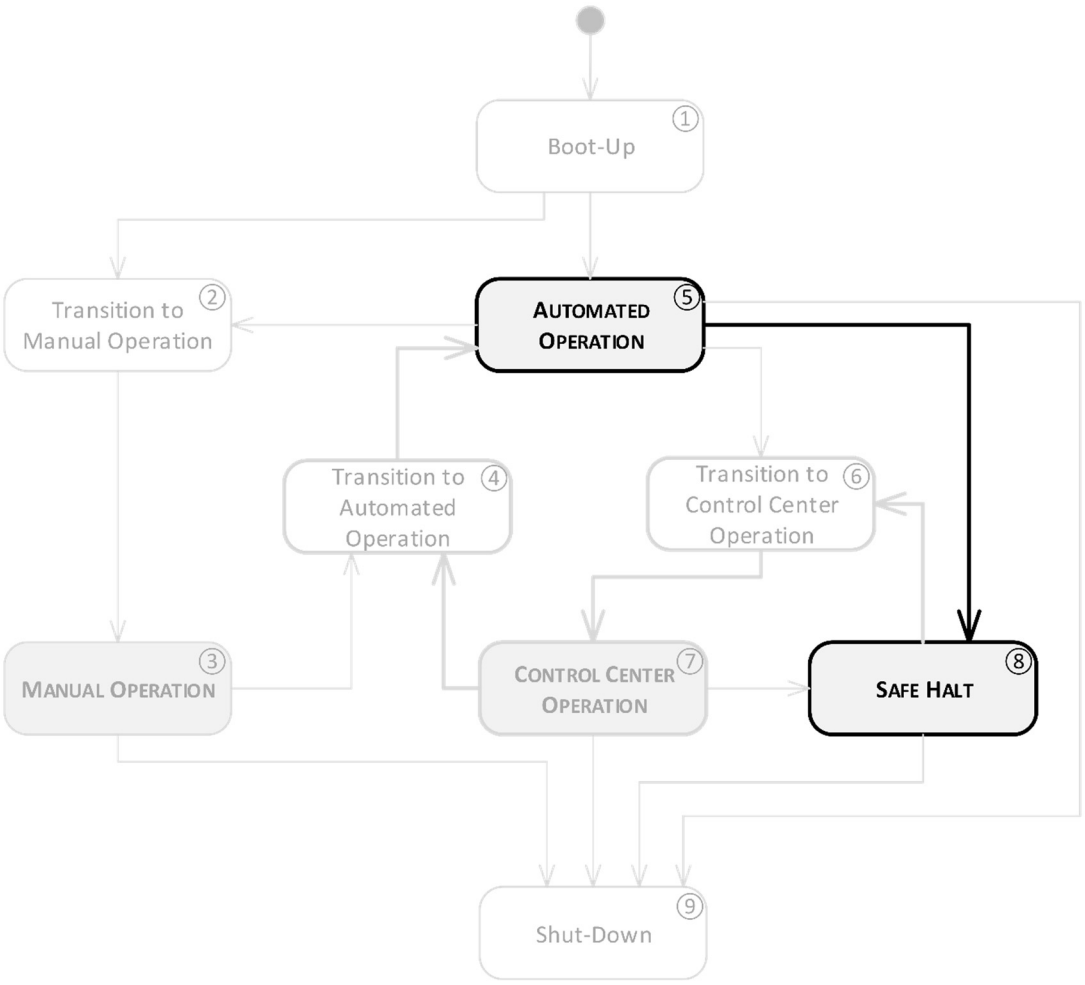

Fig. 6 Transition from automated operation into safe halt mode.

For any transition except the transition into safe halt mode, the Orchestrator reconfigures the service configuration of the vehicle. The transition into safe halt mode, however, has to be instantaneous. A reconfiguration would potentially take too much time in case of a severe degradation. For this reason, both the regular trajectory as well as the safe halt trajectory are sent to the trajectory controller and the Orchestrator utilizes its functionality to directly manipulate service interfaces to set which trajectory shall be used as input by the controller. This way, the service is kept unaware of the vehicle's operating mode. It must simply be designed with the two trajectory inputs and a way for the Orchestrator to set the active interface.

https://doi.org/10.24355/dbbs.084-202110271613-0

6.2 Transition from safe halt mode into control center operation

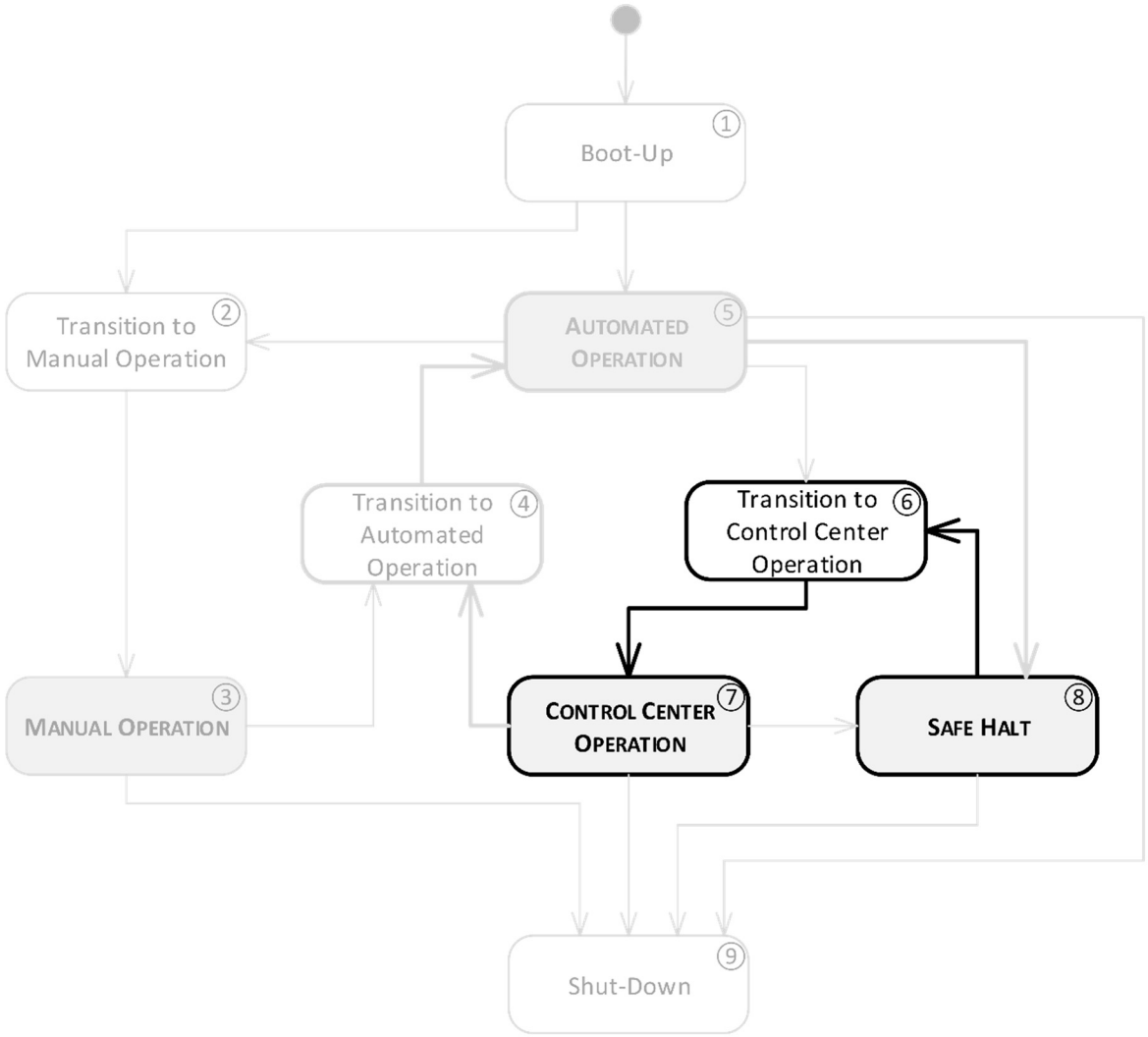

Fig. 7 Transition from safe halt mode into control center operation.

Once the vehicle has come to a standstill, the control center has to take action as any degradation severe enough to trigger the safety fallback must be assessed by a human operator. The control center is alerted that the safety fallback was triggered in a vehicle of the monitored fleet upon activation of the transition into safe halt mode. When a standstill has been reached (in the planned halt location or elsewhere), and the reaching of this state has been communicated to the operating mode management, a transition into control center operation can be initiated. As a service reconfiguration is necessary for the control center to take over responsibility for the dynamic driving task, a transitional state is entered to coordinate this reconfiguration by the Orchestrator. In order for the control center to get access to the vehicle, the active service configuration has to be reconfigured. Services have to be started to transmit data from the vehicles to the control center and to receive data from control center so the vehicle can be moved remotely. Additionally, sensors may have to be reconfigured to adjust the sensor data for transmission.

https://doi.org/10.24355/dbbs.084-202110271613-0

And of course interfaces of services providing the (preprocessed) sensor data must be connected to services for sensor data transmission to the control center. Based on the environment perception data received directly from the vehicle and the self-perception data of the vehicle that is regularly transmitted to and stored in the cloud [12] an operator in the control center will assess the problem that occurred.

The control center operator has several options of assisting the vehicle and its passengers. The operator can open a communication channel to the vehicle to communicate with the passengers, reassure, and assist them. This is especially important when children or people reliant on assistance travel in the vehicle. If the problem that occurred with the vehicle is severe and not resolvable by the operator, they can initiate having the vehicle towed to a workshop or maintenance facility and the passengers to be picked up by another vehicle. Otherwise, the operator has several options to resolve the occurred problem. If the problem was only temporary and has disappeared after the activation of the safety fallback, the operator can simply acknowledge it and thereby initiate a retransition into automated operation. If the problem appears to be location-specific (e.g., temporary degradation of localization accuracy due to environmental factors such as tree cover or high buildings), the operator can attempt to resolve the problem by moving the vehicle to a different location. In order to do this, they have two possibilities. They can request teleoperation via trajectory control or via direct control. Depending on the selected teleoperation mode, services in the vehicle have to be reconfigured differently. For teleoperation via trajectory control, the service that receives data from the control center has to be connected to the trajectory controller in the vehicle while other inputs to the trajectory controller must be disconnected. For teleoperation via direct control, the service receiving data from the control center is connected to the motion controller for direct control on the vehicle that will translate input from the control center into command for the dynamic modules of the vehicle. Additionally, an emergency trajectory will be generated by the control center and send to the vehicle to ensure that a transition into safe halt mode and an activation of the safe halt function can still be triggered.

If the problem of the vehicle could be resolved by moving the vehicle to a different location via teleoperation, the operator in the control center will acknowledge the problem as resolved and thereby initiate a retransmission into automated operation. Teleoperation may also be used to move the vehicle to a more appropriate location after a safe halt when the location the vehicle stopped at was not ideally planned by the guidance system and the vehicle is still operable by the control center.

6.3 Transition from control center operation to automated operation

If the problem that occurred with the vehicle could be resolved by the control center, the control center operator initiates a transition back to automated operation mode. Again, a reconfiguration of the service configuration of the vehicle is necessary. A transitional state is entered to coordinate this reconfiguration.

In this transitional state, the connection to the control center is terminated and all services not needed for automated operation are stopped such as services to communicate data to and from the control center. Services for sensor data processing are

https://doi.org/10.24355/dbbs.084-202110271613-0

reconfigured again to provide data in the appropriate format for environment perception services rather than a control center operator. Services for sensor data processing are reconnected to the respective environment perception services. Additionally, service interfaces of the trajectory controller are reconnected to the trajectory generation on the vehicle. Once the service reconfiguration was successful and all transitional conditions are fulfilled, the operating mode management transitions the vehicle into automated operation and the responsibility for the dynamic driving task lies with the vehicle's automated driving system again.

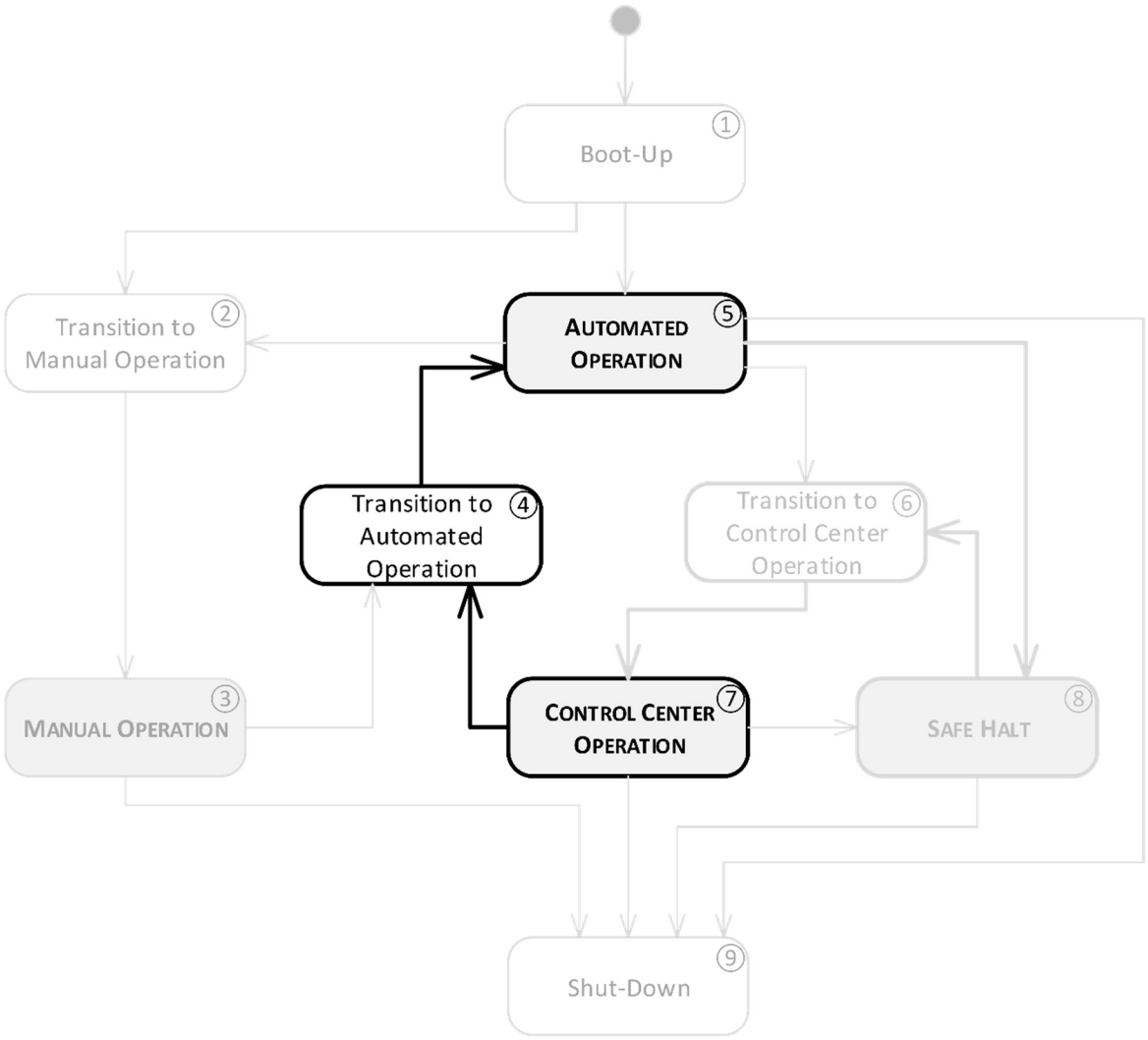

Fig. 8 Transition from control center operation to automated operation.

7 Conclusion and Outlook

In this paper, we argued the need for an operating mode management as a central decision unit for a vehicle with multiple operating modes and with the need for the coordination of non-driving related task. We demonstrated the operating mode management at the example of the operating mode management realized in the project UNICAR*agil*. It coordinates four different operating modes and the associated allocation of the responsibility for the dynamic driving task. Additionally, an operating mode

https://doi.org/10.24355/dbbs.084-202110271613-0

management is needed as a central entity to coordinate non-driving related task such as the control of an automatic door system. We presented the interaction of such an operating mode management with ASOA the Service-oriented Software Architecture. We showed its coupling with ASOA's Orchestrator as an architecture component for service discovery and orchestration and argued the contribution to a system's modularity by keeping the system's operating mode separate of the service implementation.

This operating mode management will be implemented and integrated into the four prototypes and demonstrated in operation. It will provide a valuable contribution to the overall concept of modularity pursued in the project UNICAR*agil*.

8 Acknowledgment

This research is accomplished within the project "UNICAR*agil*" (FKZ 16EMO0285, FKZ 16EMO0284K, FKZ 16EMO0290). We acknowledge the financial support for the project by the Federal Ministry of Education and Research of Germany (BMBF).

9 References

[1] Buchholz, Michael, Gies, Fabian, Danzer, Andreas, Henning, Matti, Hermann, Charlotte, Herzog, Manuel, Horn, Markus, Schön, Markus, Rexin, Nils, Dietmayer, Klaus, Fernandez, Carlos, Janosovits, Johannes, Kamran, Danial, Kinzig, Christian, Lauer, Martin, Molinos, Eduardo, Stiller, Christoph, Ackermann, Stefan, Homolla, Tobias, Winner, Hermann, Gottschalg, Grischa, Leinen, Stefan, Becker, Matthias, Feiler, Johannes, Hoffmann, Simon, Diermeyer, Frank, Lampe, Bastian, Beemelmanns, Till, Van Kempen, Raphael, Woopen, Timo, Eckstein, Lutz, Voget, Nicolai, Moormann, Dieter, Jatzkowski, Inga, Stolte, Torben, Maurer, Markus, Graf, Jürgen, Hinüber, Edgar Leuer Von, and Siepenkötter, Norbert, 2020.
Automation of the UNICARagil vehicles.
In: 29th Aachen Colloquium, Aachen, Germany.
DOI: 10.18725/OPARU-34024.

[2] Stolte, Torben, Ackermann, Stefan, Graubohm, Robert, Jatzkowski, Inga, Winner, Hermann, and Maurer, Markus, 2021.
A Taxonomy to Unify Fault Tolerance Regimes for Automotive Systems: Defining Fail-Operational, Fail-Degraded, and Fail-Safe.
In: arXiv:2106.11042 [cs, eess]. Accessed: Jun. 29, 2021. [Online].
Available: http://arxiv.org/abs/2106.11042.

[3] Kampmann, Alexandru, Alrifaee, Bassam, Kohout, Markus, Wüstenberg, Andreas, Woopen, Timo, Nolte, Marcus, Eckstein, Lutz, and Kowalewski, Stefan, 2019.
A Dynamic Service-Oriented Software Architecture for Highly Automated Vehicles.
In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. pp. 2101–2108.
DOI: 10.1109/ITSC.2019.8916841.

[4] Benali, A., Wasiak, V., and Fontaine, J. G., 2001.
Remote robot teleoperation via Internet. A first approach.
In: *Proceedings 10th IEEE International Workshop on Robot and Human Interactive Communication. ROMAN 2001 (Cat. No.01TH8591)*, Bordeaux, Paris, France. pp. 306–312.
DOI: 10.1109/ROMAN.2001.981920.

[5] Bodell, Oscar and Gulliksson, Erik, 2016.
Teleoperation of Autonomous Vehicle With 360° Camera Feedback.
Master Thesis. Chalmers University of Technology, Gothenburg, Sweden.

[6] Bruemmer, David J., Dudenhoeffer, Donald D., and Marble, Julie L., 2002.
Dynamic-Autonomy for Urban Search and Rescue.
In: *AAAI mobile robot competition*, Menlo Park, CA, USA. pp. 33–37.

[7] Bruemmer, D. J., Marble, J. L., Dudenhoeffer, D. D., Anderson, M., and McKay, M. D., 2003.
Mixed-initiative control for remote characterization of hazardous environments.
In: 36th Annual Hawaii International Conference on System Sciences, 2003. Proceedings of the, Big Island, HI, USA.
DOI: 10.1109/HICSS.2003.1174289.

[8] Appelqvist, Pekka, Knuuttila, Jere, and Ahtiaine, Juhana, 2010.
Mechatronics Design of an Unmanned Ground Vehicle for Military Applications, in *Mechatronic Systems Applications*, Di Paola, Annalisa Milella Donato and Cicirelli, Grazia, Eds. InTech.

[9] Farcas, Claudiu, Farcas, Emilia, Krueger, Ingolf H., and Menarini, Massimiliano, 2010.
Addressing the Integration Challenge for Avionics and Automotive Systems—From Components to Rich Services.
In: Proceedings of the IEEE. vol. 98, no. 4. pp. 562–583.
DOI: 10.1109/JPROC.2009.2039630.

https://doi.org/10.24355/dbbs.084-202110271613-0

[10] Kugele, Stefan, Obergfell, Philipp, Broy, Manfred, Creighton, Oliver, Traub, Matthias, and Hopfensitz, Wolfgang, 2017.
On Service-Orientation for Automotive Software.
In: *2017 IEEE International Conference on Software Architecture (ICSA)*. pp. 193–202.
DOI: 10.1109/ICSA.2017.20.

[11] Möstl, M., Schlatow, J., Peeck, J., Ernst, R., Nolte, M., Jatzkowski, I., Maurer, M., and Jankowski, A., 2018.
Controlling Concurrent Change: Managing In-Field Change in Critical Embedded Systems.
In: *2018 12th European Workshop on Microelectronics Education (EWME)*. pp. 107–112.
DOI: 10.1109/EWME.2018.8629451.

[12] Woopen, Timo, Lampe, Bastian, Böddeker, Torben, Eckstein, Lutz, Kampmann, Alexandru, Alrifaee, Bassam, Kowalewski, Stefan, Moormann, Dieter, Stolte, Torben, Jatzkowski, Inga, Maurer, Markus, Möstl, Mischa, Ernst, Rolf, Ackermann, Stefan, Amersbach, Christian, Winner, Hermann, Püllen, Dominik, Katzenbeisser, Stefan, Leinen, Stefan, Becker, Matthias, Stiller, Christoph, Furmans, Kai, Bengler, Klaus, Diermeyer, Frank, Lienkamp, Markus, Keilhoff, Dan, Reuss, Christian, Buchholz, Michael, Dietmayer, Klaus, Lategahn, Henning, Siepenkötter, Norbert, Elbs, Martin, von Hinüber, Edgar, Dupuis, Marius, and Hecker, Christian, 2018.
UNICARagil – disruptive modular architectures for agile automated vehicle concepts.
In: 27th Aachen Colloquium, Aachen, Germany.

[13] Gesetzesentwurf der Bundesregierung, 2021.
Entwurf eines Gesetzes zur Änderung des Straßenverkehrsgesetzes und des Pflichtversicherungsgesetzes – Gesetz zum autonomen Fahren.
https://www.bmvi.de/SharedDocs/DE/Anlage/Gesetze/Gesetze-19/gesetz-aenderung-strassenverkehrsgesetz-pflichtversicherungsgesetz-autonomes-fahren.pdf?__blob=publicationFile (accessed Jul. 05, 2021).

[14] Ackermann, Stefan and Winner, Hermann, 2020.
Systemarchitektur und Fahrmanöver zum sicheren Anhalten modularer automatisierter Fahrzeuge.
In: 13. Workshop Fahrerassistenz und automatisiertes Fahren, Walting, Germany.

[15] Nolte, Marcus, Jatzkowski, Inga, Ernst, Susanne, and Maurer, Markus, 2020.
Supporting Safe Decision Making Through Holistic System-Level Representations & Monitoring - A Summary and Taxonomy of Self-Representation Concepts for Automated Vehicles.
In: arXiv:2007.13807 [cs, eess]. Accessed: Aug. 05, 2020. [Online].
Available: http://arxiv.org/abs/2007.13807.

https://doi.org/10.24355/dbbs.084-202110271613-0

[16] Schräder, Tobias, Stolte, Torben, Graubohm, Robert, and Maurer, Markus, 2021.
Development of an Autonomous Family Vehicle using a Scenario-Based Design Approach.
In: 30th Aachen Colloquium, Aachen, Germany.

https://doi.org/10.24355/dbbs.084-202110271613-0